Subject: .ANDTHEN. and .ORELSE.
From: Van Snyder
Reference: 03-258r1, section 2.8.2; 04-193, 04-192, 04-357, 04-363

1 Number

TBD

2 Title

.ANDTHEN. and .ORELSE.

3 Submitted By

J3

4 Status

For consideration.

5 Basic Functionality

Provide *and* and *or* operators that are guaranteed to short-circuit evaluation.

6 Rationale

The standard presently allows a processor to short-circuit evaluation of logical expressions. For example, in `A .AND. B`, the processor is allowed not to evaluate `B` if `A` is false. It is sometimes desirable, however, to *require* that the processor not evaluate `B` if `A` is false, as opposed simply to *allowing* it not to. Here's an example:

```
    if ( present(x) .and. x /= 0 ) ...
```

One can't *depend* on the processor not trying to evaluate `x /= 0` if `x` is not present.

To support this desire, add an `.ANDTHEN.` operator, the semantics of which require the processor to evaluate the first operand first, and then prohibit it from evaluating the second operand if the first is false. The example becomes:

```
    if ( present(x) .andthen. x /= 0 ) ...
```

Similar considerations apply to the .OR. operator, leading to the desire for an .ORELSE. operator, in which the second operand is prohibited to be evaluated if the first is true.

These operators are, of course, even more useful elementally in WHERE statements and constructs. For example

```
    where ( x > 0.0 .andthen. log(x) < tol ) ...
```

The semantical property of these operators that their second operand is not evaluated if the first is false (true) could be provided by conditional expressions (04-192) or a conditional-execution intrinsic function (04-357), *viz.* `A .ANDTHEN. B` could be represented `A ? B : .FALSE.` or `IF ( A, B, .FALSE. )` and `A .ORELSE. B` could be represented as `A ? .TRUE. : B` or `IF ( A, .TRUE., B )`. Thus, if the proposal for conditional expressions proceeds, this proposal is somewhat redundant.

7 Estimated Impact

Minor. Estimated at meeting 169 to be at 4 on the JKR scale.

8 Detailed Specification

Provide *and* and *or* operators that are guaranteed not to evaluate their second operand if the first operand is false (in the *and* case) or true (in the *or* case). It is proposed that these operators be spelt .ANDTHEN. and .ORELSE.

To facilitate converting between .AND. and .ANDTHEN., and between .OR. and .ORELSE., it would be useful if the precedences of the new operators were immediately below .AND. and .OR. Since programs may already have user-defined operators with the same spelling, it would be useful if the precedence of the new operators were the same as the precedence of user-defined operators. This can be resolved later.

8.1 Suggested edits

The following suggested edits illustrate the magnitude of the project. They assume that the precedences of .ANDTHEN. and .ORELSE. are immediately below .AND. and .OR., respectively. The size of the project would not change substantially if the other decision were to prevail.

R719$\frac{1}{2}$	*andthen-op*	**is**	.ANDTHEN.	26:25+

R720$\frac{1}{2}$	*orelse-op*	**is**	.ORELSE.	26:26+

[Insert "and .ANDTHEN." after ".AND" and "and .ORELSE." after ".OR.".] 44:14

R714$\frac{1}{2}$	*andthen-operand*	**is**	[*andthen-operand and-op*] *and-operand*	120:5-6
R715	*or-operand*	**is**	[*or-operand andthen-op*] *andthen-operand*	
R715$\frac{1}{2}$	*orelse-operand*	**is**	[*orelse-operand or-op*] *or-operand*	
R716	*equiv-operand*	**is**	[*equiv-operand orelse-op*] *orelse-operand*	

R719$\frac{1}{2}$	*andthen-op*	**is**	.ANDTHEN.	120:9+

R720$\frac{1}{2}$	*orelse-op*	**is**	.ORELSE.	120:10+

[Add ", .ANDTHEN." after ".AND." and ", .ORELSE." after ".OR." in the first column of Table 7.1.] 121:7+17

[Add ", .ANDTHEN." after ".AND." and ", .ORELSE." after ".OR.".] 121:20

[Replace "Once" by "For the .AND., .OR., .EQV., and .NEQV. operators, once".] 132:4

For the .ANDTHEN. operator the second operand shall not be evaluated if the first is false. For the .ORELSE. operator, the second operand shall not be evaluated if the first is true. Otherwise, once the interpretation of an expression has been established in accordance with the rules given in 7.2.4, the processor may evaluate any other expression that is logically equivalent, provided that the integrity of parentheses in any expression is not violated. 132:8- New ¶

[Insert two new rows in Table 7.5:] 135:28+4,5+

.ANDTHEN.	Logical conjunction	x_1 .ANDTHEN. x_2	True if x_1 and x_2 are both true, but x_2 shall not be evaluated if x_1 is false
.ORELSE.	Logical inclusive disjunction	x_1 .ORELSE. x_2	True if either x_1 or x_2 is true, but x_2 shall not be evaluated if x_1 is true

[In the heading of Table 7.6, Add "x_1 .ANDTHEN. x_2" under "x_1 .AND. x_2" and "x_1 .ORELSE. x_2" under "x_1 .OR. x_2".] 136:1+2+

[In Table 7.7, replace the .OR. row] 136:5+13

Logical	.ANDTHEN.	·
Logical	.OR.	·
Logical	.ORELSE.	·